# Biased Locking in HotSpot

偏向锁适用于多线程引用的资源（object）大概率情况只被某一个线程使用的情况[1][2]，这时候访问这个object时除去第一次调用就不再需要那些原子指令了（如CAS）。当然，如果某个对象大概率被多个线程使用，这种情况就不适合再使用偏向锁，CAS等原子指令反而是负担。偏向锁的中文解释[3][7]。偏向锁默认打开，如果禁用则设置-XX:+UseBiasedLocking。

偏向锁尽量在Mark字段解决问题；轻量锁尽量用自旋锁解决问题；重量锁用系统互斥量解决问题[7][8]。

**Safepoints**

A **safepoint** is a range of execution where the state of the executing thread is well described. **Mutator** threads are threads which manipulate the JVM heap (all your Java Threads are **mutators**. Non-Java threads may also be regarded as **mutators**when they call into JVM APIs which interact with the heap)[4][5].

安全点就是Mutator线程和堆的交互是已知且规范的状态。处于安全点时，可以安全的操作堆和栈，当线程离开安全点时，就好像什么也没有发生。

可能触发安全点的场景[6]。

* Some GC phases (the Stop The World kind)
* JVMTI stack sampling methods (not always a global safepoint operation for Zing))
* Class redefinition
* Heap dumping
* Monitor deflation (not a global safepoint operation for Zing)
* Lock unbiasing
* Method deoptimization (not always)
* And many more!

1. <https://blogs.oracle.com/dave/biased-locking-in-hotspot>
2. <https://cdn.app.compendium.com/uploads/user/e7c690e8-6ff9-102a-ac6d-e4aebca50425/f4a5b21d-66fa-4885-92bf-c4e81c06d916/File/ccd39237cd4dc109d91786762fba41f0/qrl_oplocks_biasedlocking.pdf>
3. <https://www.cnblogs.com/javaminer/p/3892288.html>
4. <http://blog.ragozin.info/2012/10/safepoints-in-hotspot-jvm.html>
5. <https://medium.com/software-under-the-hood/under-the-hood-java-peak-safepoints-dd45af07d766>
6. <https://psy-lob-saw.blogspot.com/2015/12/safepoints.html>
7. <https://www.infoq.cn/article/java-se-16-synchronized/>
8. <https://www.oracle.com/technetwork/java/javase/tech/biasedlocking-oopsla2006-preso-150106.pdf>